**Z3使用教程**

**摘要**：本教程提供了对**可满足性模理论**（Satisfiability Modulo Theories，SMT）求解器Z3的介绍，并描述了其基本功能及通过Python语言API使用Z3的方法。

# 简要介绍

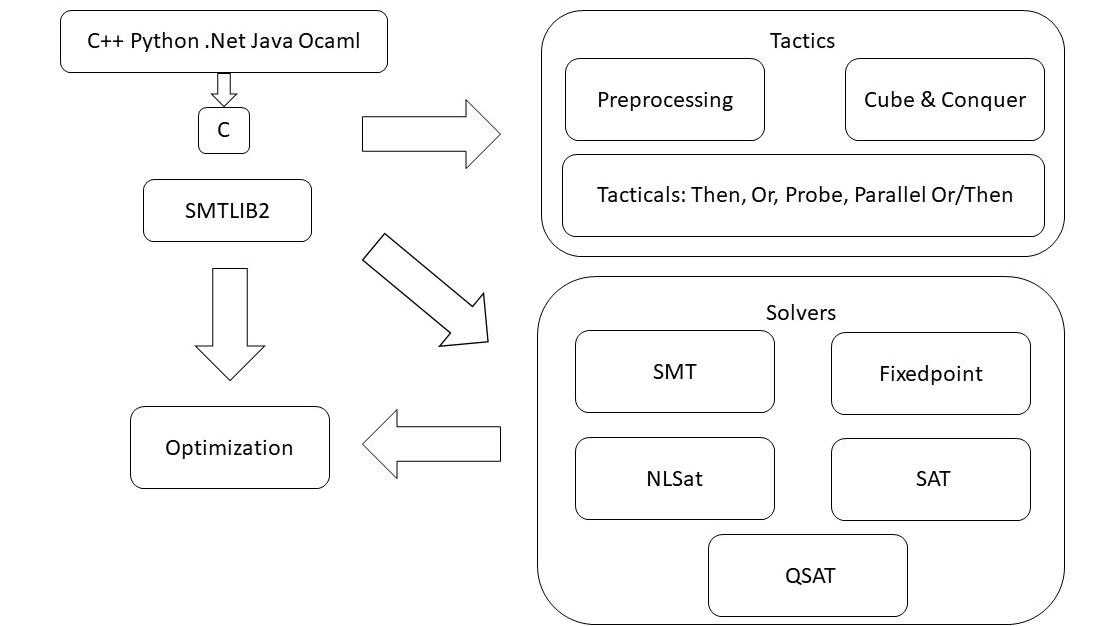
**可满足性模理论**（Satisfiability Module Theory，SMT）是指在某些背景理论（比如算术、位向量、数组和非解释函数等背景理论及其组合）下，对给定一阶逻辑公式可满足性的判断问题。我们说某个公式是可满足的（Satisfiable），当且仅当该公式存在至少一组赋值使其成真。

比如有如下公式：

1. 长度为的数组为升序排列：
2. 长度为的数组中存在值为的元素：
3. 加法交换律：
4. 一般算术：（永假式，不可满足）
5. 基本算数性质：

以上1、2公式，属于数组理论（Array Theory），定义了基本算术运算（如）以及数组的读写（如）。而3、4、5这三个公式，则属于算术理论（Arithmetic Theory），定义了比较以及加减法等运算（如）。对于上述SMT问题，可以使用SMT求解器求解。倘若公式可满足，还将输出一组令公式可满足的解。比如例子中第1个公式是可满足的，可满足的解可为且；第2个公式也是可满足的，可满足的解可为且；第3个公式是永真的；第4个公式则是不可满足的，即不存在可满足的解。

图 1：Z3系统架构



**Z3**是由微软公司开发的一款SMT求解器，上图 1显示了其系统架构。使用者可以通过SMT-LIB2脚本与Z3进行交互，这些脚本以文本文件或管道形式提供给Z3；也可以使用高级编程语言的API调用（如左上角所示），这些高级编程语言以C的API为代理进行调用。后续教程中着重于使用Python前端作为与Z3接口的方式。

**更多资源**：

1. Z3 github仓库：<https://github.com/z3prover/z3>
2. Z3 python程序示例：

<https://github.com/Z3Prover/doc/tree/master/programmingz3/code>

1. Z3编程介绍：

<http://theory.stanford.edu/~nikolaj/programmingz3.html>

# Z3的安装

**推荐第2种，方便快捷！！**

1、从源码安装（仅展示GNU/Linux系统），系统要求：GNU/Linux，比如Ubuntu。（以下例程基于Ubuntu 20.04 LTS amd64）

#安装依赖

sudo apt update

sudo apt install git make python3 python3-pip

#在Ubuntu上进行编译

git clone https://github.com/Z3Prover/z3.git

cd z3

python3 scripts/mk\_make.py --python

cd build

make

sudo make install

使用方式：

# example.py文件为待求解的脚本（也可以使用python3的交互式方式）

python3 ../examples/python/example.py

z3/examples/python/example.py为求解公式的脚本，结果中表示可满足，其中一组解为。

2、下载二进制文件、解压运行即可。

1）GNU/Linux系统，比如Ubuntu：

sudo apt update

sudo apt install python3 python3-pip

wget https://github.com/Z3Prover/z3/releases/download/z3-4.8.10/z3-4.8.10-x64-ubuntu-18.04.zip

unzip z3-4.8.10-x64-ubuntu-18.04.zip

sudo cp -R z3-4.8.10-x64-ubuntu-18.04/\* /usr/local/

使用方式：

# example.py文件为待求解的脚本（也可以使用python3的交互式方式）

python3 /usr/local/bin/python/example.py

/usr/local/bin/python/example.py为求解公式的脚本，结果中表示可满足，其中一组解为。

2）Windows系统：

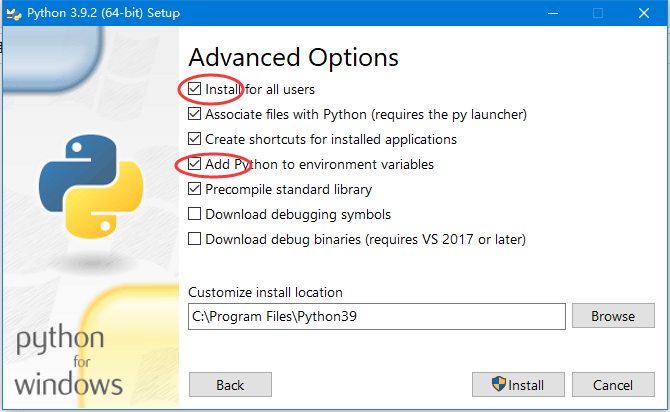
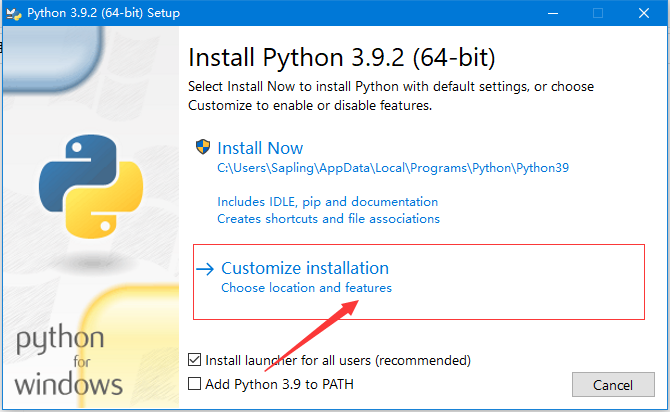
a）如果系统中已经安装python3，则跳过此步骤：

# 下载地址 https://www.python.org/ftp/python/3.9.2/python-3.9.2-amd64.exe，

（32位地址https://www.python.org/ftp/python/3.9.2/python-3.9.2.exe）

# 接着图形化安装，过程中注意下图勾选的地方

# 完成后，打开cmd键入“python -V”出现版本信息则表明安装正确



b）安装z3：

# 下载地址 https://github.com/Z3Prover/z3/releases/download/z3-4.8.10/z3-4.8.10-x64-win.zip，（32位地址 https://github.com/Z3Prover/z3/releases/download/z3-4.8.10/z3-4.8.10-x86-win.zip）

# 解压缩包，比如至C:\Program Files (x86)\z3-4.8.10-x64-win

# 然后配置PATH

a）编辑path，添加C:\Program Files (x86)\z3-4.8.10-x64-win\bin

b）新建PYTHONPATH，值为C:\Program Files (x86)\z3-4.8.10-x64-win\bin\python

使用方式（打开cmd，键入下述命令）：

# example.py文件为待求解的脚本（也可以使用python3的交互式方式）

python "C:\Program Files (x86)\z3-4.8.10-x64-win\bin\python\example.py"

example.py为求解公式的脚本，结果中表示可满足，其中一组解为。

3）MacOS系统：

# 下载地址https://github.com/Z3Prover/z3/releases/download/z3-4.8.10/z3-4.8.10-x64-osx-10.15.7.zip

# 其余步骤请参考1）

# 实例介绍

1、命题逻辑公式的基础是原子变量和逻辑连接词。以下是Z3接受的命题逻辑公式示例：

from z3 import \*

Tie, Shirt = Bools('Tie Shirt')

s = Solver()

s.add(Or(Tie, Shirt),

Or(Not(Tie), Shirt),

Or(Not(Tie), Not(Shirt)))

print(s.check())

print(s.model())

该示例求解公式，其中添加了两个布尔常量和。然后，创建一个对象并添加三个约束。调用会得出可满足性结论（即：*sat / unsat*）。该例中结果为*sat*，进而使用获取满足约束的解，结果为*False*，为*True*。为了方便起见，Z3的Python API包含一些快速求解函数，比如函数用于设置求解器、添加约束、检查可满足性并输出满足约束的解（如果*sat*的情况下）。以下程序与上述程序等价：

from z3 import \*

Tie, Shirt = Bools('Tie Shirt')

solve(Or(Tie, Shirt),

Or(Not(Tie), Shirt),

Or(Not(Tie), Not(Shirt)))

2、命题逻辑是Z3可以处理的重要但较小的公式子集，Z3还可以求解算数理论的公式，以下为示例：

from z3 import \*

x, y = Int('x'), Int('y')

solver = Solver()

solver.add(x + y == 42)

solver.add(x - 6 \* y < 2)

solver.add(x % 2 == 1)

if solver.check() == sat:

m = solver.model()

print(m[x], m[y])

该示例为求解满足公式整数解的程序。如果满足的话，则打印出整型变量的值。

3、Z3还可以求解结合了多种理论（例如数组理论和算术理论）的公式，以下为示例：

from z3 import \*

x, y, z = Ints('x y z')

A = Array('A', IntSort(), IntSort())

f = Function('f', IntSort(), IntSort())

fml = Implies(x + 2 == y, f(Store(A, x, 3)[y - 2]) == f(y - x + 1))

solve(fml)

该公式恒真，无论整数常量x、y、z，数组A，函数f为任何值/函数。注意z没有出现在公式中，但是声明中z表示一个可能用到的整数常量。

上面程序使用到的功能如下：

1）使用函数创建两个整型变量，类似的函数有；

2）用于创建类型为整型的数组；

3）用于创建从整型到整型的函数；

4）表示蕴含，类似函数有、、；

5）将由数组生成一个新数组，且新数组下标处的值为，其余部分同原数组。

此时可以手动证明上述结论。在的假设下，有、，所以将变换为，进而为，这将是一个恒等式，故上述结论成立。

4、更多示例请参考如下地址：

<https://github.com/Z3Prover/doc/tree/master/programmingz3/code>